

# Projektowanie Systemów Informatycznych

## Zajęcia 1

### Wprowadzenie

Projektowanie nowoczesnych systemów informatycznych wspierających analizę danych i procesy decyzyjne oparte na algorytmach data science

Rozgrzewka z R

## Zajęcia 2

### Cykl życia systemu informatycznego – etapy i krótka charakterystyka:

**Planowanie** (identyfikacja celów), **Analiza wymagań** (określenie funkcji systemu),  
**Projektowanie** (architektura i struktura danych), **Implementacja** (kodowanie),  
**Testowanie** (weryfikacja poprawności), **Wdrożenie** (uruchomienie systemu),  
**Eksploatacja i utrzymanie** (monitorowanie, aktualizacje), **Wycofanie** (zamknięcie systemu).

### Data Science workflow

Cykl życia procesu analizy danych w data science:

1. Zdefiniuj cel (Jaki problem staram się rozwiązać?)
2. Zgromadź dane i zarządzaj nimi (Jakie informacje są mi potrzebne?)
3. Zbuduj model (Znajdź w danych wzorce prowadzące do rozwiązań)
4. Oceń model i poddaj go krytyce (Czy model rozwiązuje mój problem?)
5. Zaprezentuj wyniki i udokumentuj je (Udowodnij, że możesz rozwiązać problem i pokaż, jak tego dokonasz)
6. Wdróż model (Wdróż model tak, aby rozwiązywał problem w środowisku produkcyjnym. Wdrażanie i utrzymywanie modelu)

## **Gromadzenie i analiza wymagań funkcjonalnych i нефункциональных w projektowaniu systemów informatycznych**

Analiza wymagań: określenie funkcji systemu

Spisanie tych wymagań pozwala na stworzenie systemu oprogramowania, który spełnia oczekiwania klienta (zamawiającego) w określonych ramach czasowych i budżetowych oraz jest zgodny z celami systemu zidentyfikowanymi w poprzednim etapie (tj. Planowanie).

**Wymagania funkcjonalne** opisują, CO system ma robić, czyli jakie funkcje ma realizować. Odpowiadają na pytania: "Co system ma umożliwić?", "Jakie zadania ma wykonywać?".

Przykłady wymagań funkcjonalnych:

- Możliwość logowania się użytkowników.
- Wyszukiwanie produktów w sklepie internetowym.
- Generowanie raportów finansowych.

**Wymagania нефункциональные** opisują, JAK system ma działać, czyli jakie ma mieć cechy jakościowe. Odpowiadają na pytania: "Jak system ma działać?", "Jakie ma mieć właściwości?".

Przykłady wymagań нефункциональных:

- Wydajność (np. czas odpowiedzi systemu).
- Bezpieczeństwo (np. ochrona danych przed nieautoryzowanym dostępem).
- Niezawodność (np. dostępność systemu przez 24/7).
- Użyteczność (np. łatwość obsługi interfejsu użytkownika).

### **Proces gromadzenia i analizy wymagań:**

1. **Identyfikacja interesariuszy:** należy zidentyfikować wszystkie osoby/grupy osób, które są zainteresowane systemem lub będą z niego korzystać.
2. **Gromadzenie wymagań:** stosuje się różne techniki odkrywania wymagań: burze mózgów, konsultacje i wywiady z kluczowymi użytkownikami, analiza dokumentów, prototypowanie.
3. **Analiza wymagań:** należy sprawdzić, czy wymagania są kompletne (niczego nie pominęliśmy), jednoznaczne (bez różnych interpretacji), spójne (niesprzeczne), testowalne (mieralne).
4. **Dokumentowanie wymagań:** wymagania są zapisywane w formie dokumentu specyfikacji wymagań, który jest podstawą do następnego etapu (tj. Projektowanie).
5. **Walidacja wymagań:** należy upewnić się, że zgromadzone wymagania są zgodne z rzeczywistymi oczekiwaniami interesariuszy. Wymagania powinny być regularnie weryfikowane i aktualizowane w trakcie trwania projektu.

## **Bibliografia**

- Farley D., Nowoczesna inżynieria oprogramowania. Stosowanie skutecznych technik szybszego rozwoju oprogramowania wyższej jakości, Helion 2023
- Śmiałek M., Rybiński K., Inżynieria oprogramowania w praktyce. Od wymagań do kodu z językiem UML, Helion 2024
- Wrycza S., Maślankowski J., Informatyka ekonomiczna. Teoria i zastosowania, PWN 2019
- Zumel N., Mount J., Język R i analiza danych w praktyce, Helion 2021